9-15-1991

# Ilona: An advanced CAI Tutorial System for the Fundamentals of Logic

Otto Mayer
*University of Kaiserslautern*

Graham E. Oberem
*Rhodes University*

Fillia Makedon
*University of Texas at Dallas*

# ILONA: AN ADVANCED CAI TUTORIAL SYSTEM FOR THE FUNDAMENTALS OF LOGIC

Fillia Makedon
Otto Mayer
Graham E. Oberem

Technical Report PCS-TR91-164

# ILONA: AN ADVANCED CAI TUTORIAL SYSTEM FOR THE FUNDAMENTALS OF LOGIC[1]

BY

OTTO MAYER[2]
GRAHAM E. OBEREM[3]
FILLIA MAKEDON[4]

Technical Report 06.89

9/15/91

---

(1) This work was funded by the Computer Learning Research Center at The University of Texas at Dallas, Box 830688, MC 3.1, Richardson, TX75083-0688, U.S.A.

(2) Department of Computer Science, University of Kaiserslautern, 6750, Kaiserslautern, West Germany.

(3) The Computer-based Education Unit, Rhodes University, Grahamstown, 6140, South Africa.

(4) Computer Learning Research Center, The University of Texas at Dallas, P.O. Box 830688, MC 3.1, Richardson, TX75083-0688, U.S.A.

# ILONA: An Advanced CAI Tutorial System for the Fundamentals of Logic

## Authors

**Otto MAYER**
Department of Computer Science
University of Kaiserslautern
6750 Kaiserslautern
West Germany

**Graham E. OBEREM**
Computer-Based Education Unit
Rhodes University
Grahamstown, 6140
South Africa

**Fillia MAKEDON**
Computer Learning Research Center
The University of Texas at Dallas
P.O. Box 830688, MC3.1
Richardson, TX 75083-0688
U.S.A.

## Correspondence

Please address all correspondence to Dr. Graham Oberem at the following address:

Dr Graham Oberem
Department of Physcis, FM-15
University of Washington
Seattle, WA 98195
United States of America

## Abstract

Mayer, O., Oberem, G.E. and Makedon, F. ILONA: An advanced CAI tutorial system for the fundamentals of Logic.

An advanced tutorial system for teaching the fundamentals of logic has been developed to run on UNIX work stations and commonly available micro-computers. An important part of this tutorial is the intelligent problem solving

environment which allows students to practise writing logical sentences in mathematical notation. A natural language system for intelligent logic narrative analysis (ILONA) allows students to type in their own logical sentences in plain English and then have the computer check their working when they write these in mathematical form. ILONA is an intelligent tutoring system which allows students a great deal of initiative in problem solving and provides a degree of flexibility in answer evaluation not found in traditional CAI systems. The concepts and structures used in the development of ILONA are easily transferable to other domains.

**Keywords**

computer assisted instruction, intelligent tutoring systems, artificial intelligence, mathematical logic, natural language processing

## Introduction

While the need for quality education in computer science is growing world wide, our ability to meet it is becoming increasingly difficult due to the shortage of qualified computer science teachers. One way to attack this "educational crisis" might be by means of the very devices that have caused it, viz., computers. After almost three decades of unfulfilled promises computer assisted instruction (CAI) is now starting to turn into a realistic hope for supporting computer science education [6].

However, the pedagogy which has been embodied in most of the CAI programs produced over the last twenty years has been that of Skinnerian stimulus response psychology. Very few programs have departed from the linear programming paradigm and few could be classified as branching in the Crowderian sense. This has resulted in programs which are of limited educational value and which many students find frustrating to use.

Two important factors are largely responsible for the renewed interest in CAI on the part of computer scientists. These are firstly the advance in software development which has been influenced to a great extent by artificial intelligence research and secondly the availability of low cost computers with increasing capabilities. It is now possible to combine the techniques of artificial intelligence with what we have learned about the medium of CAI to produce very powerful but nevertheless affordable educational software [10]. ILONA, the program discussed in this paper, falls into this new class of educational software known as intelligent tutoring systems (ITS).

## Intelligent tutoring systems

Several factors have limited the usefulness and acceptance of traditional CAI programs. For one thing, a presentation of the subject matter which does not adapt

to the needs of the individual offers little or no advantage over the use of printed materials. In order to achieve sufficient flexibility in the teaching process, the system needs to be able to deduce and maintain a detailed model of the student and use this to modify both the presentation of information and the teaching strategy as required.

An intelligent tutoring system should thus address three distinct areas of knowledge representation [11]. First, an ITS must have the basic knowledge of the area to be taught and must also be able to solve problems in this area. Secondly, the ITS must have knowledge of the student it is teaching and be able to deduce a model of that student. Thirdly, the ITS must have knowledge of how to teach and be able to develop a tutorial strategy which adapts to the needs of each particular student.

Several intelligent tutors have been built to investigate the practicality of these ideas. SCHOLAR, one of the earliest programs, had the domain knowledge encoded as a semantic network [1]. This allowed the system to diagnose misconceptions and apply appropriate tutoring. The program WHY, a meteorology tutor, further explored the idea of knowledge-driven CAI and could adjust the tutorial process on the basis of the student's progress and errors [15].

In addition simply to encoding the knowledge in a useful way, it is often important to be able to manipulate it for the purpose of problem solving. For this, rule-based expert systems are useful in that they can be made to "explain" their reasoning. MYCIN [13] is the most well-known example of a rule-based expert system which was later incorporated into a teaching program, NEOMYCIN [3].

The student model plays a key role in an ITS in providing the system with the ability to adapt to the needs of the individual. Many different approaches have been tried. Clancey [2] used an overlay model in GUIDON. Goldstein [4] used a genetic graph where the links between nodes represented the way in which learning takes place. Heines and O'Shea [5] proposed a directed graph to model the requisite skills in their rule-based tutor for learning to use the ReGIS instruction set. In these systems, the student model not only guides the tutorial process but also allows the tutoring system to diagnose misconceptions.

While the tutorial process is largely driven by the student model, it is nevertheless important to model the tutorial process itself. The ITS developed by Woolf and McDonald [20] is an excellent example of a rule-based tutor. Their system contained 40 tutoring states grouped into three levels and was used to teach the concepts of PASCAL programming. At the highest level the general approach was established. At the next level the specific teaching strategy was determined and at the lowest level this was refined into a definite tactic, for example, formulating a question to ask the student.

One of the most severe problems with traditional CAI systems has been their limited ability to analyse student input. In most cases, language processing has been

confined to keyword matching. Furthermore, it has been assumed that the programmer would be able to predict all possible student inputs and make allowance for these. Clearly, this is an impossible task and the rather unimpressive language processing capabilities of conventional CAI programs have led to student discontent in many cases. Few ITS programs have incorporated natural language processing capabilities. One example is the program ACE [14] which could analyse complex written interpretations of NMR (nuclear magnetic resonance) spectra typed in by chemistry students. ACE could cope with multiple concepts expressed in a single input and could make allowances for misspellings and abbreviations.

The processing capabilities of microcomputers have now progressed to the point where it is possible to implement efficient language processing systems on them. An example of one program which runs on a microcomputer and which has natural language capability is ALBERT, a physics problem solving monitor and coach. ALBERT allows students to type in physics problems in plain English and is able to engage the student in a natural language dialog to help him/her to solve the problem [9]. In this case, the domain of interest was sufficiently limited to enable an efficient language system with a vocabulary of manageable size to be constructed. ALBERT not only understands the problems typed by the students but is also able to solve them, and can monitor and coach students while they produce solutions of their own.

In ILONA, the advanced CAI system we describe here, an environment is provided for the students to formulate logical sentences and practise representing them in mathematical form. ILONA is an intelligent tutoring system which runs on commonly available microcomputers, specifically the Macintosh series of computers and IBM PC compatibles. Many of the elements found in other intelligent tutoring programs are present in ILONA and have been made to operate efficiently in this context.

## The Fundamentals of Logic

Not all topics are equally suitable for presentation via the medium of the computer. In some cases, the complexity of the domain is such that a multi-media approach is demanded if the students are to understand the concepts fully. In other cases, the skills being taught may be such that another medium is more appropriate [19]. However in most subjects, students are required to practise problem solving and this is often an activity which can be managed well by the computer.

In the domain of logic, the concepts are sufficiently abstract to make presentation via the computer feasible, but not so complex as to make the instructional design process prohibitive. Patrick Suppes and his co-workers at Stanford have offered a very successful computer-taught class in elementary logic since 1972 [16]. This course consists of 29 computer lessons which form a comprehensive introduction to the subject. Suppes reports a positive response to the course on the part of his

students and comments favourably on the degree of flexibility provided by the computer in presenting it [17].

The process of problem solving in this area is such that it can be encoded as a finite set of rules or an algorithmic computer program. Suppes has worked on automatic theorem provers for teaching elementary logic and axiomatic set theory and has used these in his computer-based lessons [18]. Our emphasis has been on building a problem solving monitor for student use.

The basic rules of logic have been intuitively understood for more than 2000 years in jurisprudence, mathematics, and philosophy. Logic was developed as a mathematical discipline about one hundred years ago, and in the last few decades it has become one of the corner stones of technological advance through the development of computers. It governs almost all applications of computer science and advanced communications.

It is essential for the design of circuits that make up computers and communication systems, as well as for the design and verification of computer programs and communication protocols. A basic knowledge of logic should therefore be part of one's education just as the basics of physics, chemistry, and mathematics are. Due to their clear definition and formal nature the fundamental rules of logic are extremely well suited to being taught by means of CAI.

ILONA consists of two sections. The first is a tutorial in which the basic concepts of logic are presented. After studying this the student should be able to identify the atomic expressions of logic called propositions (they represent sentences which are true or false) and to understand

- how they are connected
- how they can be equivalent and
- how they can be quantified.

The second section is an intelligent problem practise environment. Here the student is encouraged to formulate logical sentences and to translate them into mathematical form. He/she can use the practise system to perfect his/her skill in representing such sentences both in every-day language and in formal mathematical notation.

The introduction to the fundamentals of logic and the practise environment are presented in such a way that the program can even be used at the senior high school level.

**The Logic Tutorial**

The tutorial part is split into several sections and subsections which introduce the basic building blocks of logic, the so-called propositions, and combinations of these

by the most frequently used logic operators, namely, conjunction, disjunction, negation, and implication. Logical equivalence is discussed extensively in terms of truth tables and illustrated by means of simple logic circuits. In addition, some elements of first order logic are presented. Predicates are then introduced as propositional forms and their quantification by existential and/or universal quantification is discussed. This corresponds roughly to the first two lessons in the Stanford curriculum [16].

While this tutorial is fairly traditional in its approach, every effort has been made to engage the student in an interactive learning experience. Information and new concepts are interspersed with thought provoking questions and exercises which allow the student to test his/her understanding of the content and which help to reinforce the ideas being presented. Figure 1 shows an example of the type of presentation and interaction employed in the tutorial section of the lesson. Figure 2 illustrates the use of the mouse pointing device for student input in the section on equivalence.

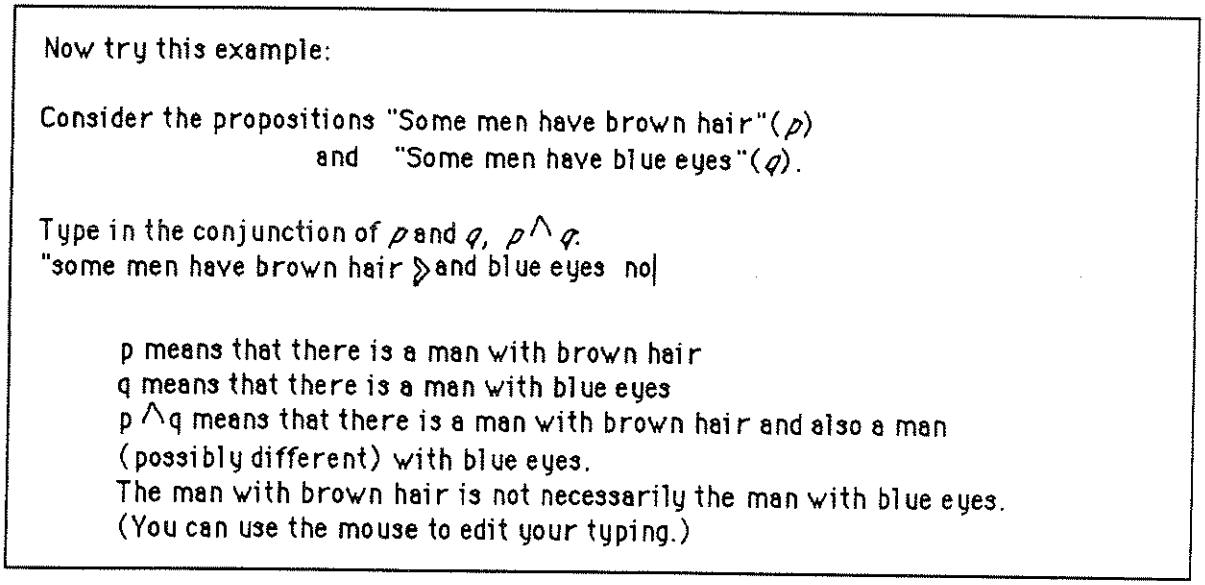


Figure 1. An example of a student exercise in conjunction.

## The Intelligent Problem Practise Environment

When students have completed the tutorial section of the lesson, an opportunity is provided for them to practise what they have learned in an intelligent problem solving environment. This represents a significant departure from the traditional approach to computer-based problem solving in the degree of flexibility and control which the student is given.

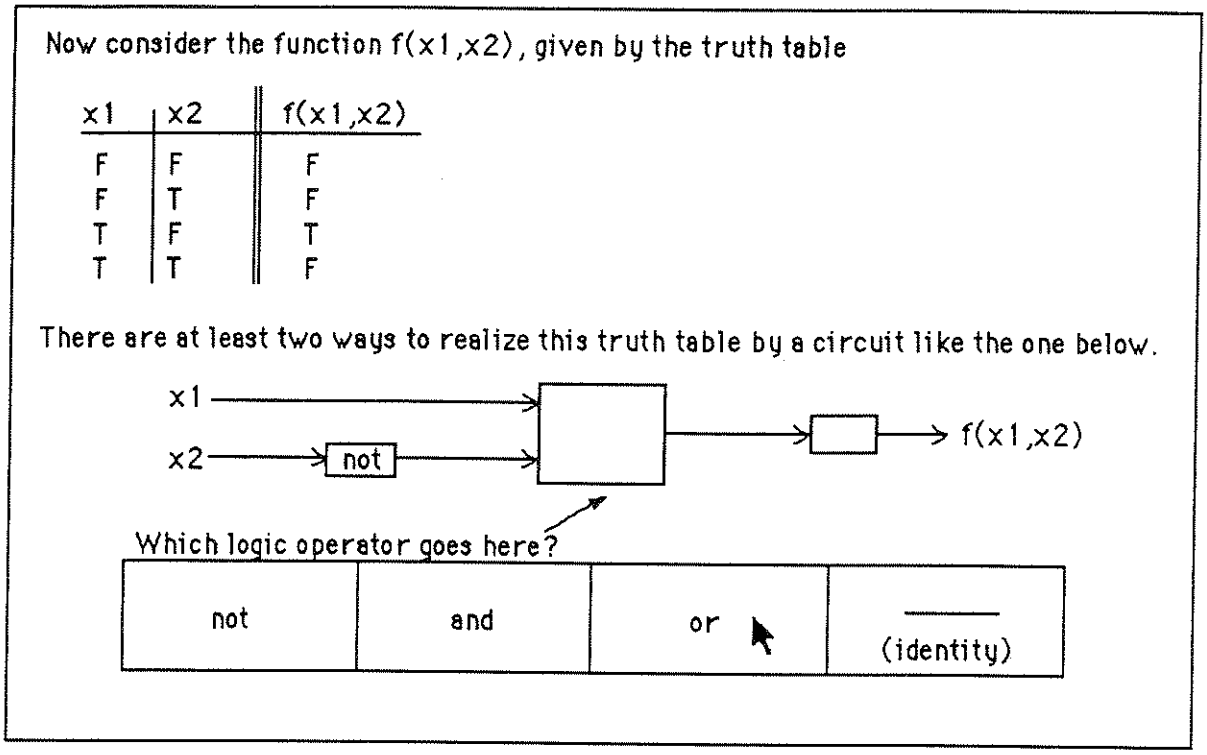


Figure 2. An example of a student exercise with mouse input.

An important part of the problem practise environment is the Intelligent LOgic Narrative Analyzer, after which ILONA is named. This is a program which has the ability to "understand" logical sentences which the user types in in plain English and which is capable of translating them into mathematical form. It is further able to compare solutions which it generates with those produced by the student. It can describe its working and give specific feedback about errors which the student makes. Also, when the student is unable to proceed, relevant and specific help is offered.

The logic narrative analyser has been incorporated into ILONA's intelligent tutoring system, the architecture and basic design concepts of which are discussed below.

Figure 3 shows the components of the problem practise environment of ILONA.

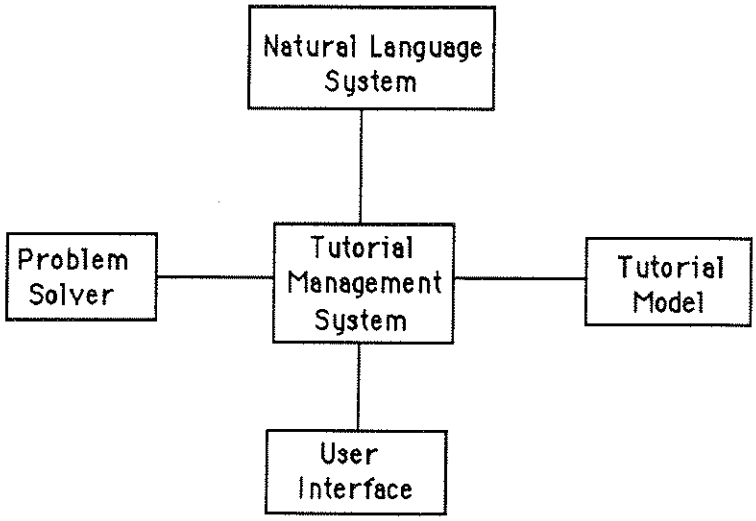


Figure 3 -- Components of ILONA's problem practise environment.

*The Natural Language System*

The natural language system used in ILONA is derived from the one developed for ALBERT [8]. It works on the principle of pattern recognition. Words in the lexicon are grouped into semantic categories and a number is associated with each category. Each word in the student input is looked up and converted to the number representing the semantic category of that particular word in this context. The result is an overall bit pattern representing the full input. This pattern is searched for semantically significant syntactic sub-patterns. When a sub-pattern is matched, the appropriate semantic sub-routine is used to extract the useful information from that part of the input sentence. The syntactic sub-patterns may vary in length from three to seven words. This language system is database driven and it can be moved from one domain to another by loading a new vocabulary and a new set of syntactic patterns.

In the rather limited domain of propositional calculus, few syntactic rules (sub-patterns) are needed as the sentence structures found are standard and simple. However, the vocabulary needed is extensive. A feature of ILONA's version of the language system which was not a part of the original is its ability to cope with words which are not in the lexicon. Consider the sentence

> "If it is the case that all big hilogies are zygolistic, then some hilogudgies are zygolistic."

Although we do not know what "hilogies" or "hilogudgies" are nor what "zygolistic" is, we can infer from the sentence structure that the first two words are very likely to be nouns and that the other one is most likely some property which those nouns might exhibit. To make allowance for this, the language system in

ILONA inserts a given bit pattern for unknown words and uses appropriate semantic pattern templates to try to "recognize" such words in the context of the words around them. This type of fuzzy parsing turns out to work very successfully for ILONA as detailed information about nouns and their properties are not required if all we need to do is to convert the input sentence into mathematical form.

The natural language system in ILONA takes as input sentences of the form shown in Table 1. Its output is a frame of the type proposed by Minsky which contains information about how the sentence was composed [7]. This knowledge is represented in such a way that it can be used by other parts of ILONA.

It is not the case that the sun is green.
No man exists who can fly.
If some dogs are brown and white then some cats are grey or white.
All men can sing and shout implies some women can dance and play.
Birds have wings and cats have tails.
Every man can write his name and some girls sing loudly.
etc.

Table 1 - Examples of sentences which ILONA can understand.

*The Problem Solver*

The problem solver takes as its input the data generated by the natural language system. These data are checked to ensure that they are complete and that a sensible solution is possible. Figure 4 shows an example in which an implication was incomplete. If the data are faulty or incomplete, the problem solver aborts and the student must correct the input sentence before proceeding. This is also the case for sentences which contain syntactic structures with which ILONA is unable to cope.

```
Type in a logic proposition:  If the sun is shining then the

Your sentence could not be analysed by ILONA.
You did not complete the implication.
Press RETURN and type a different sentence.
```

Figure 4 - An Incomplete Implication

The problem solver has two overall requirements. First it needs to generate a mathematical representation of the problem which can be checked against the

student's version thereof. Secondly it needs to be able to write out its solution with an explanation of what it is doing.

In principle, the student can use any variable name and any function name he/she chooses to represent the objects and properties in the input sentence. In ILONA the choice has been limited to some extent but nevertheless, it is unlikely that the student's solution will match ILONA's exactly. It is therefore necessary for ILONA to generate a solution which is generic in terms of the variable and function names. Since the answers can be quite lengthy, the internal representation must be such that rapid comparison with the student's answer is possible. To achieve this, the final answer is represented as a single text string with one character per answer element. This allows answer checking by a single line in the source code rather than a slow looping element by element operation. This can regarded as a pattern matching operation since a bit pattern of the answer as a whole is used to match with the student input.

ILONA can describe its working by running through the final answer an element at a time and interpreting each in sequence as shown in Figure 5. Figure 6 shows another example of a problem which ILONA has solved, this time with reflexivity.

Type in a logic proposition: it is not the case that all men can swim and there exists a women who can run and jump

ILONA would have worked it out like this:

Let the variable "x" represent an element of " men"
Let the function "f()" represent the property, ability or quality associated with " swim".
Let the variable "y" represent an element of " women"
Let the function "g()" represent the property, ability or quality associated with " run".
Let the function "h()" represent the property expressed by " jump".

Therefore: $\neg ( \forall x f(x) ) \wedge \exists y ( g(y) \wedge h(y) )$

Figure 5 - A problem which ILONA has solved.

Type in a logic proposition: If it is the case that some dogs are brown and black and some cats are grey then some mice are grey and some mice are brown

ILONA would have worked it out like this:

Let the variable "x" represent an element of " dogs"
Let the function "f()" represent the property, ability or quality associated with " brown".
Let the function "g()" represent the property expressed by " black".
Let the variable "y" represent an element of " cats"
Let the function "h()" represent the property, ability or quality associated with " grey".
Let the variable "u" represent an element of " mice"

Therefore: $\exists x ( f(x) \wedge g(x) ) \wedge \exists y\, h(y) \rightarrow \exists u\, h(u) \wedge \exists u\, f(u)$

Figure 6 - A problem with reflexivity.

*The User Interface*

The user interface in ILONA is designed to give the student as much flexibility as possible while constraining him enough to make the answer analysis feasible. A complicating factor is the need for the student to input several special symbols such as the notation for "and" and "or". It was therefore decided to use the mouse as an input device and to provide, in addition to the special symbols, a range of variable and function names for the student to use. The symbols provided for student use are shown in Figure 7. The student constructs his/her answer by clicking successively on the symbols he/she wishes to use.

Construct your answer by clicking on the table below.
Click on "done" when your answer is complete.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| erase | x | u | f() | h() | m() | s() | P | P2 | done |
| | y | w | g() | k() | n() | t() | Q | Q2 | |
| | $\forall$ | $\exists$ | $\vee$ | $\wedge$ | $\rightarrow$ | $\neg$ | ( | ) | |

Figure 7 - Symbols for student input via mouse.

The student may use the variable and function names provided in any order and in principle this could make answer analysis difficult. Accordingly a system is provided in ILONA whereby the student's input is translated into an internal representation which matches that generated by the problem solver, i.e., a text string with a generic one character per answer element representation.

Provision is also made in the user interface for the student to (i) erase symbols which he/she has used, (ii) make requests for help and (iii) abort the current problem.

*The Tutorial Model*

The tutorial model is the least extensively developed part of ILONA. It has two main functions, viz., answer checking and providing help. The model has not been developed as a rule-based tutor, an approach often cited in the literature.

When the student clicks on "done", the internal representation of his/her answer is compared with that generated by ILONA. If a mismatch is found, a specific comment is provided about where the error occurs. This is illustrated in Figure 8.

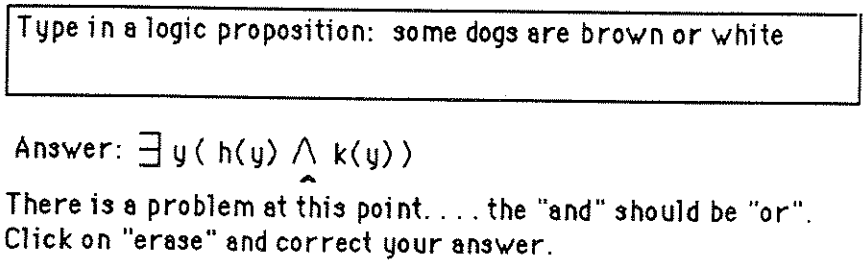


Figure 8 - Feedback on incorrect student input.

The student may then click on "erase" and correct the problem.

If the student requests help, a menu item at the top of the screen, a specific hint is given about what the next element of input should be. The tutorial model obtains this information by interrogating the problem solver. This is illustrated in Figure 9. If help is used too often, a suitable comment is made when the problem has been completed.

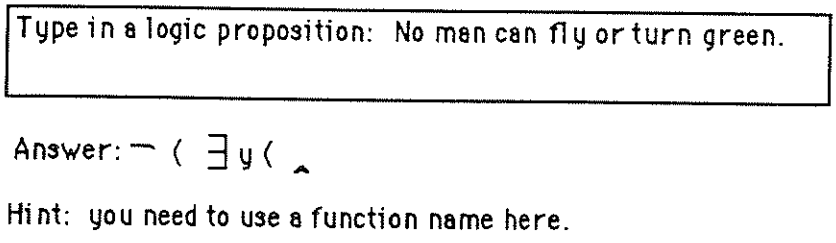


Figure 9 - ILONA responds to a request for help.

*The Tutorial Management System*

The tutorial management system may be regarded as the mediator in the tutorial process. It directs control to the appropriate part of the program or to the user depending upon what action is required next.

In ILONA the normal flow of activities is from the user interface to the natural language system, to the problem solver, back to the user interface, and then to the tutorial model which may interrogate the problem solver in order to provide relevant feedback. If help is requested, the tutorial model accesses the problem solver to obtain a hint on how to proceed and the information is fed to the student via the user interface.

*Scope of ILONA's Expertise*

Of course there are some limitations on the complexity of input sentences which can be handled successfully by ILONA. In order to be analyzable an input sentence must exhibit an intrinsic structure as described by the following BNF notation:

<analyzable sentence> ::= <comp. proposition>
| <comp. proposition> —> <comp. proposition>
<comp. proposition> ::= <proposition>
| <proposition> <cd-operator> <proposition>
<proposition> ::= <quantifier><variable><predicate>
| ¬<quantifier><variable><predicate>
| <atomic proposition>
| ¬<atomic proposition>
<predicate> ::= <atomic predicate>
| <atomic predicate><cd-operator><atomic predicate>
<co-operator> ::= ∧ | ∨
<quantifier> ::= ∀ | ∃
where atomic proposition represents an indivisible proposition and atomic predicate represents an indivisible predicate.

To understand, what this BNF notation means, we read "::=" as "is " and "|" as "or". Then an analyzable sentence is either a compound proposition, or an implication of a compound proposition to a compound proposition. A compound proposition is either a proposition, or a conjunction or disjunction of propositions. A proposition is a quantified predicate, a negated quantified predicate, an atomic proposition, or a negated atomic proposition. A predicate is understood to be either an atomic predicate, or a conjunction or disjunction of atomic predicates.

## Conclusions

This advanced CAI tutorial was written using the cT authoring language developed at Carnegie Mellon University [12]. The cT system allows a program to execute on an IBM PC compatible computer, a Macintosh computer, or an advanced work station such as the SUN 3/60 or the IBM R/T without modification to the source code. This makes ILONA completely transportable across several delivery platforms.

Our system may be used as an introduction to the fundamentals of logic in any basic mathematics or computer science course at the high school or undergraduate university level. The only prerequisite is that the student should have an understanding of the concept of a function as mapping from an argument domain into a value domain.

ILONA, the intelligent practise environment, may be regarded as an intelligent tutoring system in that it incorporates a natural language system, an expert problem solver and a degree of flexibility in answer judging not found in conventional CAI programs. However, the tutorial model has not been extensively developed to date and a fully developed student model, something usually associated with intelligent tutoring systems, is not present. It would be useful to track student progress and to develop a student profile based on specific errors which he/she makes or does not make. ILONA can be made to do this by recording the details of each session. This provides useful data for the diagnosis of student difficulties and these data can be used in the design of the student model.

With the power of desktop computers today and their relatively low cost, the development of tutoring systems with natural language interfaces, problem solving ability, and a useful degree of flexibility in tutorial management has become feasible. The natural language component of ILONA is easily transferable to other domains and we expect that tools of this kind to support the efficient development of advanced tutoring systems will become widespread in the next few years.

## Acknowledgement

This work was funded by the Computer Learning Research Center at The University of Texas at Dallas.

## References

[1] Carbonell, J.R. (1970), AI in CAI: An artificial intelligence approach to computer aided instruction. *IEEE Transactions on Man-Machine Systems, 11,* pp. 190-202.

[2] Clancey, W.J. (1979), *Dialog management for rule-based tutorials.* Heuristic Programming Project report No. HPP-79-9. (Stanford University, CA).

[3] Clancey, W.J. and Letsinger, R. (1981), *NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching.* Heuristic Programming Project report No. HPP-81-2. (Stanford University, CA).

[4] Goldstein, I.P. (1982), The genetic graph: a representation for the evolution of procedural knowledge. In D. Sleeman and J.S. Brown (eds), *Intelligent Tutoring Systems*, pp. 51-78. (Academic Press, London).

[5] Heines, J.M. and O'Shea, T. (1985), The design of a rule-based tutorial, *International Journal of Man-Machine Studies, 23,* pp 1-12.

[6] Maurer, H., and Makedon, F. (1987) *COSTOC: Computer Supported Teaching of Computer Science.* Proc of IFIP 3rd International conference on Remote Education and Informatics: Teleteaching '86, North Holland vol. 1987, pp. 93-106.

[7] Minsky, M. (1975), A framework for representing knowledge. In P. Winston (ed), *The Psychology of Computer Vision.* (McGraw-Hill, New York).

[8] Oberem, G.E. (1986), *ALBERT: An Intelligent Computer-based Tutor for Elementary Mechanics Problems.* Ph.D. Thesis. (Rhodes University, Grahamstown, South Africa).

[9] Oberem, G. E. (1987), *ALBERT: a physics problem-solving monitor and coach.* Proc. of Conference on Computers in Post-Secondary Education: Learning in Future Education. The University of Calgary, Alberta, Canada, pp. 179-184.

[10] O'Shea, T. and Self, J. (1983), *Learning and teaching with computers -- artificial intelligence in education.* (The Harvester Press, Brighton, Sussex)

[11] Self, J.A. (1974), Student models in computer aided instruction, *International Journal of Man-Machine Studies 6,* pp. 261-276.

[12] Sherwood, B.A. and Larkin, J.H. (1989), New tools for courseware production. *Journal of Computing in Higher Education, 1,* pp. 3-20.

[13] Shortcliffe, E.H. (1976), *Computer-based Medical Consultations: MYCIN.* (American Elsevier, New York, NY).

[14] Sleeman, D. and Hendley, B.A. (1982), ACE: A system which analyses complex explanations. In D. Sleeman and J.S. Brown (eds), *Intelligent Tutoring Systems,* pp. 99-118. (Academic Press Inc., London).

[15] Stevens, A. and Collins, A.M. (1977), *The goal structure of a Socratic tutor.* Tech. Rep. No. 3518. (Bolt, Beranek and Newman, Inc., Cambridge, MA).

[16] Goldberg, A. and Suppes, P. (1976), Computer-assisted instruction in elementary logic at the university level. *Educational Studies in Mathematics 6*, pp. 447-474.

[17] Suppes, P. and Sheenan, J. (1981), CAI course in Logic. In P. Suppes (Ed.) *University-Level Computer-Assisted Instruction at Stanford: 1968-1980*, pp. 193-226. (Institute for Mathematical Studies in the Social Sciences, Stanford University.)

[18] Suppes, P. (1984), *The next generation of interactive theorem provers*. In R.E. Shostak (Ed.), 7th International Conference on Automated Deduction, Napa, California, Lecture Notes in Computer Science 170, pp. 303-315. (New York: Springer-Verlag.)

[19] Trollip, S.R. (1986), *Common pitfalls when implementing computer-based training*. Proc. of Computer-based education and Training Seminar, pp 125-136. (Control Data, Johannesburg, South Africa).

[20] Woolf, B. and McDonald, D.D. (1984), Building a computer tutor: design issues, *Computer 17*, pp. 61-73.

---

**About the authors**

**Dr Otto Mayer** is a professor of computer science at the University of Kaiserslautern, West Germany, where he has established an active CAI laboratory. He is the author of a CAI course entitled "Introduction to Programming in LISP" which is used successfully at various universities in Europe and North America and has also produced several other CAI lessons for computer science.

**Dr Graham Oberem**, a professor of physics, has been involved in the development and use of CAI lessons for more than a decade. He is director of the Computer-Based Education Unit which he established at Rhodes University, South Africa. His intelligent physics tutor, ALBERT, has gained a lot of international recognition due to its ability to understand physics problems stated in plain English and its use of a natural language dialog interface for student input.

**Dr Fillia Makedon** is a professor of computer science and director of the Computer Learning Research Center which she established at the University of Texas at Dallas. She has participated in several international CAI projects and used CAI courses in her teaching of computer science. Her primary research interest is in the visualization of algorithms.